ISSN: 2473-4901
v9 n5973

An IoT Based New Platform for Teaching Tiny Machine Learning

Juefei Yuan
jyuan@semo.edu

Sam Elfrink
selfrink3s@semo.edu

Zhouzhou Li
zli2@semo.edu

The Deparment of Computer Science
Southeast Missouri State University
Cape Girardeau, MO 63701, U.S.

Kedai Cheng
kcheng@unca.edu
The Deparment of Mathematics and Statistics
University of North Carolina at Asheville
Asheville, NC 28804, U.S.

Qiuyu Han
2003138@hlju.edu.cn
Heilongjiang University
Harbin, Heilongjiang, CN

Abstract

The widespread presence of sensor-rich intelligent edge devices, such as mobile phones and IoT gadgets, is evident in our daily lives. By integrating artificial intelligence (AI) with these devices, we can unlock a myriad of real-world applications, from smart homes and retail solutions to autonomous driving and beyond. However, state-of-the-art deep learning AI frameworks/ systems often require massive resources (e.g., large labeled datasets, high-performance computational resources, and many AI experts) for training and testing. This hinders the applications of these powerful deep learning AI systems on edge devices. In this paper, we introduce an economical and user-friendly web application platform, termed ESP32-CAM, into the coursework. This platform is characterized as an Internet of Things (IoT) device. Furthermore, we integrate a Tiny Machine Learning (TinyML) – based neural network model, which is optimized for the limited computational capabilities of IoT devices while minimally impacting the performance, into this device with the specific objective of detecting human emotions. The integration of this platform and knowledge of TinyML offers students ample opportunities for hands-on practice within the classroom, fortifying their practical skill set. Additionally, it serves as a catalyst for sparking their curiosity and engagement in learning about other AI-related disciplines or topics, including robotics, autonomous driving, 3D scene retrieval, and more.

Keywords: Tiny Machine Learning, Internet of Things, Artificial Intelligence, Human Emotion Recognition, Face Recognition.

1. INTRODUCTION

The rising demand for low-power, low-cost, and high-performance devices capable of executing intelligent tasks like object recognition, voice recognition, and predictive maintenance (Ji Lin, et al, 2022) has driven advancements in various technological realms. Notably, Tiny Machine Learning (TinyML), a sub-category of Artificial Intelligence, has emerged as a direct response to these requirements. These devices include smart cameras, remote monitoring devices, wearable devices, audio capture hardware, various sensors, and more. Additionally, advancements in hardware and software technologies have made it possible to develop efficient machine learning models that can run on small devices with limited memory and processing power.

TinyML is the intersection of different technology areas and drivers, it sits at the juncture between Internet of Things (IoT) devices (Peña-López, 2005) (ITU-T, 2012), machine learning and edge computing, and it is progressing rapidly because of the combination of multiple drivers (Partha Pratim Ray, 2022).

TinyML is promising to deliver an efficient and effective way for Embedded Systems and Internet-of-Things (IoT) units, which take into account the computation, memory, and energy constraints of the devices (Simone Disabato, et al, 2020)(Ji Lin, et al, 2020). All of these devices typically contain sensors, processors, and communication hardware that allow them to collect and exchange data with other devices, applications, or cloud-based systems. They can be used to collect and analyze human behaviors or personal information data in real time, allowing for more efficient and effective decision-making in a wide range of industries.

Another driving force behind the development of TinyML is the need for privacy and security in the age of IoT. By deploying machine learning models on edge devices, data can be processed locally without the need for cloud connectivity, reducing the risk of data breaches and cyber attacks (Mauro Conti, et al, 2018) (Frahim J, et al, 2015) (Servida F, et al, 2019).

Figure 1 illustrates the scope of TinyML within the broader context of Artificial Intelligence (AI). AI forms the overarching umbrella under which Machine Learning (ML) resides, with Deep Learning (DL) nested as a subset within ML. TinyML emerges as a specialized branch of machine learning that is fine-tuned for operation on resource-constrained devices, such as those in the IoT domain. It draws its roots from deep learning. Through the integration of TinyML, IoT devices are empowered to execute intelligent functions locally by utilizing the data they gather, negating the requirement for significant computational capabilities.

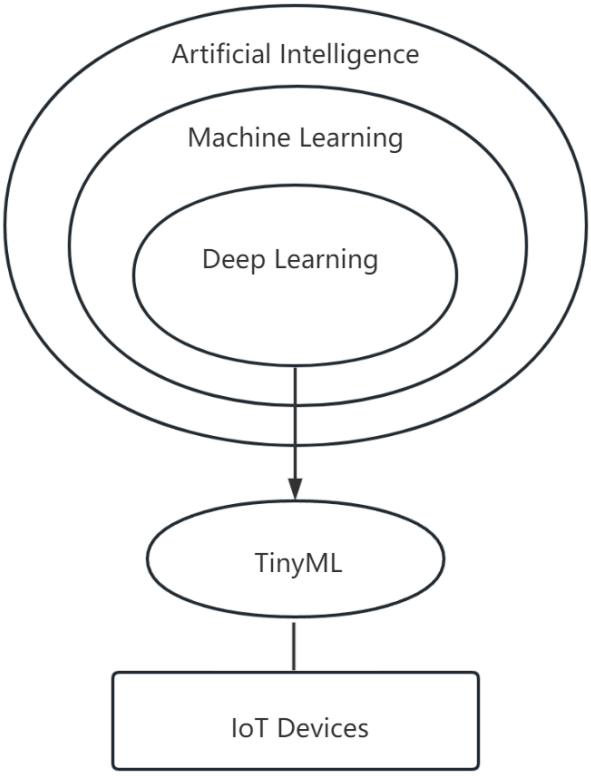

Figure 1: The Context of TinyML

Unfortunately, when taking into account cost, performance, and efficiency, finding an appropriate platform for students to practice during their coursework proves to be a challenge. The major concerns include:

- The widely-used Convolutional Neural Networks (CNNs), including ResNet, InceptionNet, and VGG, necessitate devices with substantial computational capabilities.
- IoT devices, including smart home appliances, possess constrained computational power. They are not equipped to support CNN models that demand substantial resources, such as high-performance computing capabilities.
- Implementing traditional machine learning/ deep learning methods on IoT devices is a challenging task, as it is

difficult to do so without compromising a significant amount of performance.

A well-constructed platform is essential for students to effectively practice TinyML techniques on IoT devices with constrained computational capabilities.

The subsequent sections of this paper are structured in the following manner. In the 'Literature Review' section, we review the benefits of combining TinyML with IoT devices. In the 'Background' section, the Internet of Things (IoT) device-based human emotion recognition platform will be introduced, and its advantages will be explained. In the 'Teaching Objectives' section, the educational aims of the Tiny Machine Learning course will be explored. Following that, a compilation of exercises will be presented to bolster the teaching objectives through the utilization of the platform. The "Conclusions and Future Work" section will offer a wrap-up of the findings and shed light on potential avenues for future research and course development.

2. LITERATURE REVIEW

The benefits of tiny machine learning are numerous, and research in this area is ongoing. One study published in the SIGCSE TS 2022 developed a tiny machine learning educational tool TinyMLedu. The aim of TinyMLedu is to establish a global network of researchers and practitioners who are dedicated to advancing the field of TinyML in developing countries. Additionally, the organization strives to create and disseminate high-quality educational resources that are open-access and accessible to people worldwide (Brian Plancher, et al, 2022) (Harvard Tiny Machine Learning Open Education Initiative, 2022).

Another study published in the IEEE Internet of Things Journal evaluated the performance of tiny machine learning algorithms for human activity recognition on an embedded device. The results showed that the algorithms achieved high accuracy while requiring minimal computational resources and memory, making them suitable for deployment on low-power devices (Yang, et al, 2021).

The third study published in the Sensors Journal developed a low-power hardware platform for implementing a tiny machine learning system for real- time fall detection. The results showed that the system achieved high accuracy and low power consumption, making it suitable for deployment in wearable devices for the elderly (Kwon, et al, 2020).

In conclusion, the widespread presence of sensor-rich intelligent edge devices, including billions of mobile phones and IoT devices, is prominent in our everyday lives (Egham UK, 2017) (Frahim J, et al, 2015). The integration of Artificial Intelligence (AI) with these edge devices opens up a plethora of real-world applications spanning smart homes, smart retail, autonomous driving, and beyond. However, deploying cutting-edge deep learning AI frameworks/systems is resource-intensive, often necessitating large labeled datasets (Olga Russakovsky, et al, 2015), high-performance computing resources (NVIDIA, 2023), and a substantial pool of AI experts for model training and evaluation (Haoyu Ren, et al, 2021). This poses a barrier to implementing these potent deep learning AI systems on edge devices. Our TinyML initiative is geared toward enhancing the efficiency of deep learning AI systems by minimizing the computational demands, reducing the number of engineers involved, and utilizing a small amount of data, all while maintaining performance. This is aimed at fostering the burgeoning market of edge AI and the Artificial Intelligence of Things (AIoT) (Alexander S. Gillis, 2022).

3. BACKGROUND

ESP32-CAM, as shown in Figure 2, is a Web App built on IoT hardware, offering swift, precise, and cost-efficient Human Emotion Recognition. A representative use case/scenario is provided below:

- Initially, new users must register their facial image with the Web App. Subsequently, a unique ID is allocated to the associated face. Once the facial image is stored in the system, the user no longer has access to it.
- The ESP32-CAM hardware can be positioned in locations conducive to capturing human faces, such as near doors, on walls, or atop desks.
- A user stands in front of the ESP32-CAM hardware. The Web App processes the input, which encompasses the user's entire body, to pinpoint the location of the face. Subsequently, it analyzes the facial expressions and provides an emotion recognition prediction for the user.

The ESP32-CAM hardware is priced around $8.00. When taking into account the ancillary cables and bridge devices, the total expenditure for a single

ISSN: 2473-4901
v9 n5973

set of practice hardware amounts to approximately $15.00.

The Web App is primed for deployment by a user within just 1 minute. Additionally, it boasts a cold start time of 10 seconds, outpacing all known platforms in startup speed. It's also worth highlighting that the Web App is fully open-source, eliminating the need for instructors to be concerned about potential Intellectual Property issues. Moreover, it's easy to maintain and expand, as ultimately, there are only four source files to be maintained, two of which are header files.

In addition to its cost-effectiveness and commendable performance, another appealing aspect of this Web App is its ability to integrate IoT with Artificial Intelligence (AI) methodologies, including Tiny Machine Learning (TinyML) techniques

Figure 2: Views of the ESP32-CAM Board from the Front and Back

Internet of Things

IoT is an emerging technology, and despite the keen interest shown by many students, a course covering IoT fundamentals is usually available only as an elective in degree programs. By incorporating exercises that utilize IoT devices into the TinyML course, cutting-edge and valuable material can be infused into this course that otherwise focuses on established techniques. This approach renders it unnecessary for students to enroll in a full-fledged IoT course to gain exposure to embedded hardware and wireless communication.

This IoT hardware-based platform is also beneficial for students engaging in research on edge computing, a trending subset of cloud computing, with an emphasis on tailored computation for delivering swift and precise outcomes. The deep learning model, which was trained on a dataset comprising various human emotions using high-performance computing systems, demonstrated impressive efficacy and its versatility was substantiated (Xiaofeng Lu, 2022) (Ankan Bhattacharyya, et al, 2021). Nevertheless, when this model is implemented on an IoT device like the ESP32-CAM, there are shortcomings in terms of recognition accuracy and processing speed, signifying an opportunity for further research and optimization. In one of the capstone projects, a group of students discovered that the ESP32-CAM could not accommodate deep learning models with an extensive parameter set, and any attempt to streamline the model by cutting down the number of parameters led to a severe decline in recognition accuracy and response time.

Tiny Machine Learning

As previously mentioned, a TinyML model has been incorporated into the ESP32-CAM web application. This iteration of TinyML represents an emerging technology that entails the deployment of machine learning models, including streamlined versions of deep learning models, onto devices with limited resources. The learning methodologies can be classified into supervised, semi-supervised, and unsupervised learning, with human emotion recognition falling under the category of supervised learning. A majority of Computer Science and Electrical & Computer Engineering departments have already incorporated Machine Learning and Deep Learning into their course offerings. As a result, this TinyML course can provide practical experiences for students, allowing them to apply and integrate the knowledge they have acquired from the Machine Learning/Deep Learning courses in a holistic manner. The interactive nature of this TinyML model elicited strong enthusiasm among the students, particularly in relation to the human emotion recognition application.

This platform, integrated with TinyML, is also advantageous for students to undertake research in transfer learning. In transfer learning, the existing TinyML model is not modified; instead, a new learning framework is constructed atop the pre-existing model. For instance, it can be employed to extend human emotion recognition capabilities to include body language recognition.

Additionally, with the ongoing advancement and proliferation of machine learning, deep learning, and Tiny Machine Learning models, students have a wealth of opportunities to keep pace with cutting-edge research in domains like 3D object detection, analysis of 3D environments both indoor and outdoor, self-driving cars, among

ISSN: 2473-4901
v9 n5973

others, leveraging the widespread yet computationally constrained capabilities of IoT devices.

4. TEACHING OBJECTIVES

Teaching objectives and outcomes that students should be able to achieve upon completion of this course:

1. Understanding of Core Concepts:
   - Students should gain a solid understanding of the fundamental concepts of Machine Learning, Deep Learning, Tiny Machine Learning, and Internet of Things (IoT).
   - Students should be familiar with the terminology and theory behind human emotion recognition.
2. Working with IoT Devices:
   - Students should be able to configure and use IoT devices, specifically platforms like ESP32-CAM. The control panel of the ESP32-CAM web app is depicted in Figure 3.
   - Students should understand how these devices can be integrated with sensors and actuators.
3. Model Development and Optimization:
   - Students should be proficient in developing machine learning models suitable for emotion recognition.
   - Students should be able to optimize these models for deployment on resource-constrained IoT devices (TinyML).
   - Students should know how to perform transfer learning to enhance or modify existing models for new applications.
4. Data Handling and Processing:
   - Students should be adept at collecting, processing, and handling datasets, particularly those related to human emotions.
   - Students should understand the importance of data preprocessing and be able to employ techniques to clean and prepare data for training.
5. Implementation and Deployment:
   - Students should be capable of implementing TinyML models on IoT devices.
   - Students should know how to deploy a model within an IoT ecosystem, ensuring efficient communication between devices.
6. Performance Evaluation:
   - Students should be able to evaluate the performance of the emotion recognition models.
   - Students should be familiar with metrics and methods to analyze model accuracy, speed, and efficiency, especially in the context of TinyML.
7. Ethical and Privacy Considerations:
   - Students should be aware of the ethical implications and privacy concerns associated with emotion recognition and IoT devices.
   - Students should understand the responsibilities and best practices in handling sensitive data.
8. Problem-solving and Innovation:
   - Students should be able to identify potential challenges and limitations in implementing TinyML on IoT devices for emotion recognition and develop innovative solutions to address these issues.

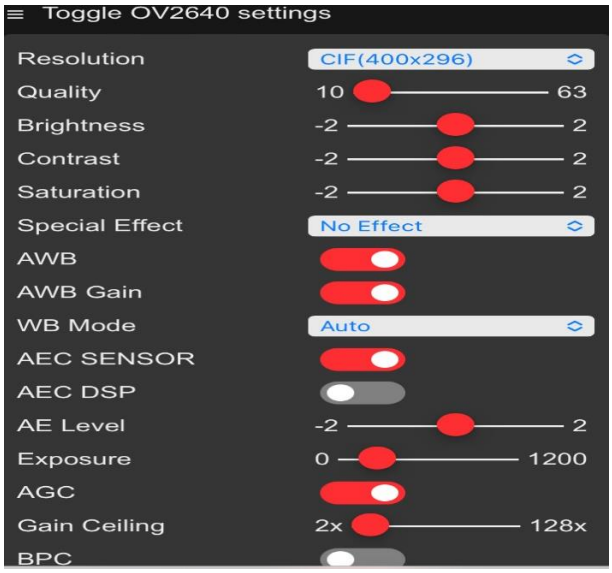

Figure 3: Control panel of the ESP32-CAM web app

By achieving these objectives, students will be well-equipped to engage in real-world applications and research in Tiny Machine Learning, especially in the context of IoT devices and human emotion recognition.

5. EXERCISES

Incorporating practical exercises in the course will enable students to apply the concepts they learn and gain hands-on experience. There are 10 independent exercises in total that can be

included in the course and can be categorized into 4 sections

1. The initial pair of exercises focuses on acquainting students with the fundamentals of IoT and TinyML. This involves setting up IoT devices and engaging in data collection and preprocessing.
    a. Setting Up an IoT Device: Students will set up an IoT device, such as a Raspberry Pi or an Arduino board, connecting various sensors to it (like temperature sensors or cameras).
    b. Data Collection and Preprocessing: Using the setup from the previous exercise, students will collect a dataset (e.g., environmental readings, images, or audio snippets) and learn preprocessing techniques suitable for TinyML.

2. Subsequently, the following duo of exercises delves into model development and refinement. Students will construct a basic emotion recognition model and learn how to optimize it for TinyML.
    c. Building a Basic Emotion Recognition Model: Students will utilize a dataset of facial expressions or voice modulations to train a rudimentary emotion detection model using common machine learning tools.
    d. Optimizing the Model for TinyML: Using model compression and quantization techniques, students will refine their emotion recognition model to make it lightweight and suitable for deployment on low-resource IoT devices.

3. Once the students have grasped the intricacies of TinyML model design, three exercises centered on deployment and performance assessment is presented. These exercises encompass model deployment on IoT devices, performance evaluation, and an introduction to transfer learning.
    e. Deploying the Model on an IoT Device: Students will take their optimized emotion recognition model and deploy it to the IoT device from the first exercise, converting the model to the appropriate format if necessary.
    f. Evaluating Model Performance: In a real-world environment, students will evaluate how well their deployed model performs in recognizing emotions, considering accuracy, speed, and resource utilization.
    g. Introduction to Transfer Learning for TinyML: Using a pre-trained model, students will learn how to adapt it to a new but related task by fine-tuning with a smaller dataset, an essential technique for TinyML given limited resources.

4. The following three exercises pivot toward real-world applications and security considerations. This includes exercises such as real-time emotion detection, understanding security and privacy within IoT, and undertaking innovative projects among others.
    h. Real-time Emotion Detection System: Building on prior exercises, students will create a system where the IoT device can detect and respond to emotions in real-time, such as changing LED colors based on detected emotions.
    i. IoT Security and Privacy: Students will explore potential vulnerabilities in their setups, understanding common attack vectors in IoT. They will then implement basic security measures to protect data and ensure privacy.
    j. Innovative Project Design: In a group or individually, students will conceptualize and begin the development of a unique application of TinyML on IoT devices, incorporating all they have learned. This could be a smart home application, a health monitoring system, or any other innovative idea.

An ancillary project has been crafted to cater to the interests of several students who expressed a desire to engage in related research within this TinyML course, conduct an independent study, or incorporate it into their capstone course.

In summary, by engaging in these exercises, students will not only understand the theoretical concepts but also gain practical skills in applying Tiny Machine Learning in real-world IoT applications, particularly in the field of human emotion recognition.

Figure 4 shows an enumeration of the hardware, Integrated Development Environment (IDE), and software that are utilized in the exercises:

ISSN: 2473-4901
v9 n5973

Hardware:

- a ESP32-CAM IoT board (including a mini camera)
- a FTDI Mini USB to TTL Serial converter
- a mini-USB cable
- selfies

IDE:

- Arduino IDE with the ESP32 add-on

Software:

- TensorFlow
- Visual Studio Code
- Vi
- Gunzip
- Wireshark
- Cscope
- OWASP ZAP
- Microsoft STRIDE
- Gcc

Figure 4: Curriculum Resource Requirements

Data collection and preprocessing

In the teaching materials for this course, images depicting human emotions are sourced from Flickr (FLICKR, 2023) and Google Images (GOOGLE, 2023). These images are classified into two categories, namely neutral and smiling. Figure 5 and Figure 6 showcase two sets of images, each containing 10 examples from the respective categories. The data for each category is divided into training and testing subsets, with 75% allocated for training and the remaining 25% designated for testing. Table 1 provides an overview of the composition of the training and testing data.

Figure 5: Example neutral human emotions images in our dataset.

Figure 6: Example smiling human emotions images in our dataset.

Images	Neutral	Smiling
Training	407	524
Testing	135	174
Total	542	698

Table 1: Training and testing information of our human emotion dataset.

Once the data collection is complete, the first step in preprocessing involves filtering out images that are evidently not relevant, for instance, removing images that do not fall under either the neutral or smiling categories. Subsequently, data augmentation techniques are employed to expand the dataset. This includes employing geometric transformations and random erasing among other techniques.

In addition, the face landmark localization information is acquired through the utilization of a light weight Human Face Detection Model (MTMN) (ESP-FACE, 2019), which is based on MobileNetV2 (Mark Sandler, et al, 2018) and Multi-task Cascaded Convolution Network (MTCNN) (Kaipeng Zhang, et al, 2016). The MTNN is capable of generating five face landmarks, encompassing information on ten coordinates corresponding to the left eye, left corner of the mouth, tip of the nose, right eye, and right corner of the mouth. Figure 7 shows the workflow of MTNM.

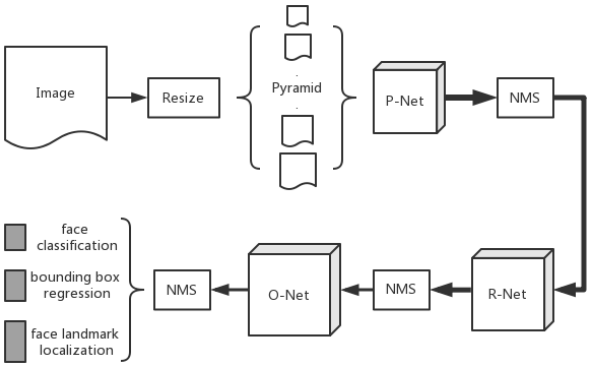

Figure 7: Workflow of MTNM.

Furthermore, in order to assemble the training dataset (which necessitates a significantly larger quantity of images than what can be manually captured by the ESP32-CAM's built-in camera), we modified the ESP32-CAM code to accommodate 1,240 (542 neutral & 698 smiling) pre-saved face images stored on a micro SD card, and to output the five face landmarks for each of these face images.

ISSN: 2473-4901
v9 n5973

Once we have obtained the five face landmarks for each image, we proceed to calculate the lengths of ten segments connecting these face landmarks; for instance, the segment length between the left and right eye. Figure 8 illustrates the ten segments derived, with P0 to P4 representing the five face landmarks, and Len0 to Len9 denoting the ten segments.

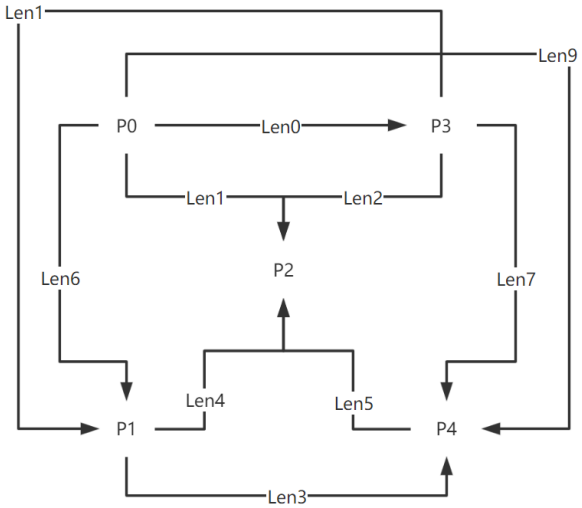

Figure 8: 10 segments connecting the 5 landmarks.

Considering that each individual's facial landmarks, as well as the segment lengths between these landmarks, vary, we normalize this segment data to establish consistency. To achieve this, we revise Len1 through Len9 employing the formula Len[n] = Len[n] / Len[0], where n ranges from 1 to 9.

Conclusively, we extract the lengths of nine segments for each face image and input this data into our TinyML model.

This section presents an excellent opportunity for students to learn the ropes of data collection and preprocessing.

Developing Machine Learning Model

TensorFlow (TENSORFLOW, 2023) is the framework that we utilize in this course to develop the our TinyML model. It is an open-source machine learning framework developed by the Google Brain team. It was initially released in 2015 and has since become one of the most widely used libraries for developing machine learning and deep learning applications. TensorFlow is a popular choice for both newcomers and experienced practitioners in the field.

To accomplish this exercise, students will need to build a fully connected neural network (NN) model using TensorFlow first, which generally involves the following steps:

1. Install TensorFlow: Before starting, the students need to ensure they have TensorFlow installed in their working environment.
2. Import Libraries: The students need to import TensorFlow and any other necessary libraries such as Numpy, Matplotlib, etc.
3. Load and Preprocess Data: Load the image dataset we collected. Typically, datasets are divided into three parts: training, validation, and testing. The students also need to normalize the data, for example by scaling the pixel values of images to the range [0, 1].
4. Define the NN Architecture: The students need to create the structure of the Neural Network. This includes adding input layers, hidden layers, and output layers. Select the number of neurons for each layer and choose activation functions for each layer (such as ReLU, sigmoid, tanh, etc.).

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 32)                320

 dense_7 (Dense)             (None, 32)                1056

 dense_8 (Dense)             (None, 1)                 33

 activation_2 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,409
Trainable params: 1,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: An example of a neural network model structure.

In summary, these four steps involve setting up the development environment by installing TensorFlow, importing necessary libraries, loading and preprocessing the data to be used, and defining the architecture of the neural network which outlines how it will process the data. An example of a neural network model structure is illustrated in Figure 9.

Machine Learning Model Training & Performance Assessment

After defining the neural network architecture, students have essentially laid the groundwork for their model. Students have set up the input, hidden, and output layers, and defined the activation functions that will help their network learn complex patterns. Now, before the students proceed to train the model with data, it's essential

to establish how the model will learn from this data and how its performance will be measured. This is where compiling the model comes into play, and it's crucial as it sets up the learning process. Here the students will define the loss function to measure how well the model is performing, and an optimizer which is the algorithm that the model will use to improve itself. Essentially, compiling bridges the gap between the architecture the students have defined and the training phase that comes next. Here below are the steps that the students need to follow:

1. Install Compile the Model: Before training, the students need to compile the model. This step involves specifying the loss function (which measures how far the predictions are from the true values), the optimizer (which is the algorithm used to perform the weight updates), and the metrics that the students want to track.
2. Training the Model: Train the model using the training data. Choose the number of epochs (iterations over the entire dataset) and the batch size (the number of samples that are used to calculate a single update of the weights).
3. Model Evaluation and Tuning: Evaluate the model on a validation set. If the performance is not satisfactory, the students may need to tweak the architecture or hyperparameters, or consider techniques like dropout or batch normalization to improve performance.
4. Testing the Model: Once the students are satisfied with the model's performance on the validation set, perform a final evaluation on the test set.
5. Make Predictions: Use the trained model to make predictions on new, unseen data, e.g., testing data.
6. Save the Model: After training and evaluating, save the model so it can be reused later without retraining.

```
Epoch 298/300
25/25 [==============================] - 0s 6ms/step - loss: 0.3976 - acc: 0.8118 - val_loss: 0.4690 - val_acc: 0.7944
Epoch 299/300
25/25 [==============================] - 0s 6ms/step - loss: 0.4064 - acc: 0.7970 - val_loss: 0.4718 - val_acc: 0.7944
Epoch 300/300
25/25 [==============================] - 0s 6ms/step - loss: 0.4037 - acc: 0.8051 - val_loss: 0.4771 - val_acc: 0.7742
```

```
1 test_acc = model_proto.evaluate(X_test,  y_test, verbose=2)
2 print(test_acc)
```

```
8/8 - 0s - loss: 0.3774 - acc: 0.8266 - 52ms/epoch - 6ms/step
[0.37743696570396423, 0.8266128897666931]
```

Figure 10: An example of a neural network model structure.

In summary, these steps are centered around training the neural network, assessing its performance, making adjustments if necessary, using it for predictions, and saving the final product for future use. Figure 10 shows the loss values and accuracies for the training, validation, and testing datasets of the NN model after 300 epoch training. It is evident from the results that the trained NN model achieves a prediction accuracy of 82.6% on the testing dataset, which comprises data that the model has not seen before.

Transformation of a Fully Connected Neural Network into an Optimized Tiny Machine Learning Model

Converting a Fully Connected Neural Network into an Optimized Tiny Machine Learning (TinyML) Model involves several steps to ensure that the model is lightweight and efficient enough to run on resource-constrained devices like ESP32-CAM. Below is the information for each step:

1. Pruning: Begin by pruning the neural network. This means removing neurons or weights that have little to no effect on the model's performance. Pruning helps in reducing the size of the model without significantly affecting its accuracy.
2. Quantization: Convert the weights and, possibly, activations from floating-point to a lower bit-width integers (e.g., 8-bit integers). This process reduces the model size and speeds up inference but might incur a slight drop in accuracy.
3. Knowledge Distillation: Optionally, students can use knowledge distillation where the students train a smaller model to imitate the behavior of the larger, trained model. This way, the smaller model learns to mimic the performance of the larger model but with fewer parameters.
4. Fine-tuning: After pruning and quantization, it is often beneficial to fine-tune the model further with a smaller learning rate to recover any loss in performance that might have occurred due to the size reduction techniques.
5. Model Conversion: Convert the optimized model to a format that is compatible with the target platform. For example, TensorFlow Lite is a popular format for TinyML models.
6. Optimize for the Target Platform: Some platforms have specific constraints or features that can be leveraged for further optimization. Tailor the model for the specific characteristics of the target hardware, which is the ESP32-CAM device we use in this course.
7. Benchmark and Validate: Before deploying, it's crucial to benchmark the TinyML model to

ISSN: 2473-4901
v9 n5973

ensure that it meets the performance and resource requirements of the target device. Validate that the model's accuracy is still within acceptable limits.

Optimizing for TinyML is often an iterative process and might require several rounds of optimization and validation to achieve the desired balance between size, speed, and accuracy. Figure 11 illustrates a comparison between the sizes of the original Neural Network model and the converted TinyML model. From the data presented in Figure 11, it is evident that the size of the original Neural Network model is 63kb, whereas the TinyML model is significantly smaller at 5kb. This substantial reduction in size makes the TinyML model much more appropriate for deployment on IoT devices.

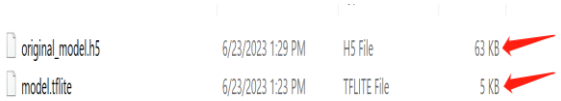

Figure 11: A comparison between the sizes of the original Neural Network model and the converted TinyML model.

Deploy the human emotion recognition Web application to an ESP32-CAM IoT board & Performance Assessment

Students will be required to acquire the skills necessary to deploy a web application onto an IoT device.

1. The application code is set up within the Arduino IDE (Arduino, I., 2019), following the installation of the necessary add-on for the ESP32 board.
2. Students are required to link the ESP32-CAM board to a host machine (where the Arduino IDE is operational), then proceed to cross-compile the code within Arduino and transfer the executable from the host to the board.
3. Once reset, the board initiates with the human emotion recognition web application prepared and operational.
4. Navigate to a specified URL to access the application's control panel, where a user can register a face and subsequently assess if the board is capable of accurately identifying the user's emotion.

In this course, teams are composed of 4 or 5 members each. The hardware required for the exercises includes an ESP32-CAM board, an FTDI Mini USB to TTL Serial converter, and a mini-USB cable.

To execute the TinyML model on the ESP32-CAM IoT board, it is necessary to transform the TensorFlow Lite model (model.tflite) into a C/C++ source file (model.cc). The ESP32-CAM IoT board incorporates libraries that facilitate this conversion. Figure 12 depicts the libraries needed, along with the commands to perform the conversion.

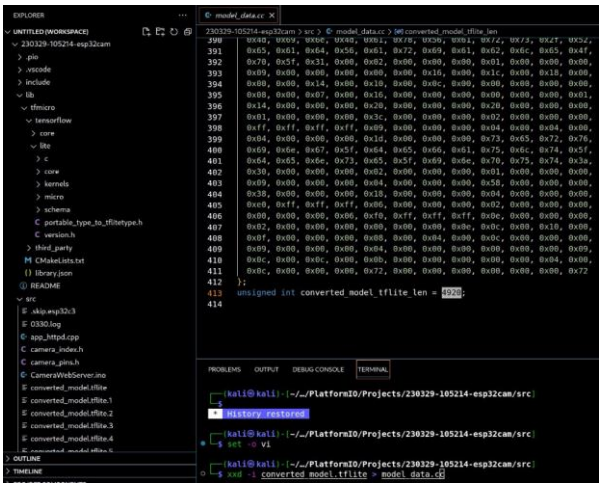

Figure 12: Conversion of TensorFlow Lite Model to C/C++ Source File on ESP32-CAM.

We evaluated the model utilizing a test dataset comprising 135 neutral facial expressions and 174 smiling faces. The model achieved an overall accuracy of 79%, which remains the same performance as the large-size machine learning model. Figure 13 and Figure 14 depict the prediction results for a neutral face and a smiling face respectively. In real-time testing scenarios, when the ESP32-CAM IoT board camera captures a human image or detects a person, it computes the lengths of nine segments between the five identified landmarks and conducts predictions.

It's imperative to note that during the training phase, labels were assigned to the facial expressions: neutral faces were labeled as 0, and smiling faces as 1. Consequently, the prediction value for a neutral face should approximate 0, while for a smiling face, it should be in proximity to 1. For instance, in Figure 13, the prediction value for the test image with a neutral face is 0.09, closely aligned with the expected value of 0. Similarly, in Figure 14, the test image with a smiling face has a prediction value of 0.97, which is near the anticipated value of 1.

For a majority of the students, this marks their initial encounter with an IoT device or an embedded system, leaving them both intrigued and apprehensive. A well-structured and detailed

ISSN: 2473-4901
v9 n5973

instruction manual can play a pivotal role in facilitating the prompt completion of this exercise, instilling confidence in the students as they learn novel techniques, skills, or methods.

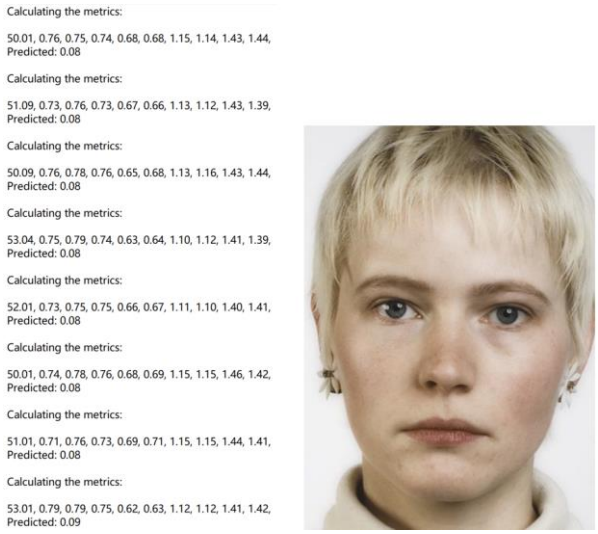

Figure 13: An example of prediction results for a neutral face in real time

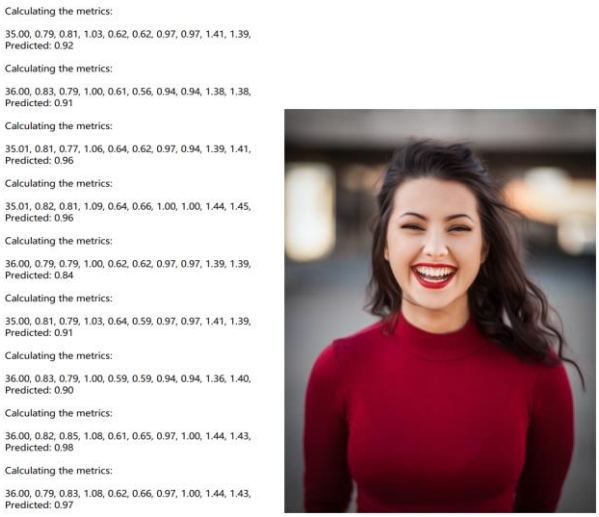

Figure 14: An example of prediction results for a smiling face in real time

This exercise enables students to delve into the rudimentary aspects of Tiny Machine Learning and the development and deployment of IoT applications. In addition, they get to familiarize themselves with the essential components of a web application, which hold the potential to be deployed on diverse hardware. Although students might have been exposed to web application development concepts during their first or second year, for most, this represents the first instance of deploying a web application on standalone physical hardware.

Moreover, this serves as an excellent opportunity for students to gain hands-on experience with IoT within the context of a Tiny Machine Learning course.

Monitor and Maintain the Deployed Model

Monitoring and maintaining the Model is a critical phase in the machine learning lifecycle, particularly once the model is deployed in a production environment. It entails continuously observing the model's performance to ensure that it is still aligned with the desired outcomes and not degrading over time. This may involve tracking various metrics and comparing them against benchmarks or thresholds. Monitoring is especially important because the data the model encounters in the real world may evolve, causing what is known as "model drift". When a significant discrepancy or degradation in performance is noticed, the model may require maintenance. This can include fine-tuning, retraining with fresh data, or even redesigning the model to adapt to the new data patterns. Additionally, monitoring can help in identifying any unexpected behavior or anomalies which could indicate issues with the data, the model, or the production environment. Regular maintenance ensures that the model remains robust, accurate, and reliable as the dynamics of the data and environment change.

6. CONCLUSIONS AND FUTURE WORK

In this course, the IoT device employed is the ESP32-CAM. This device is a versatile and compact camera module equipped with an ESP32 chip. The ESP32-CAM is particularly well-suited for applications in IoT due to its integration of Wi-Fi and Bluetooth capabilities, as well as its support for image capturing and processing. The module's small form factor and low power consumption make it an ideal choice for projects that require real-time image processing, especially when deployed in environments with resource constraints.

The use of the ESP32-CAM in this course facilitates the hands-on learning of Tiny Machine Learning in a practical context. Its capacity for capturing and processing images in real-time is invaluable for emotion recognition exercises. Furthermore, its integration capabilities with other sensors and devices over Wi-Fi and Bluetooth add layers of complexity and possibilities, allowing students to develop multifaceted projects. By working with the ESP32-CAM, students gain practical experience and insights into the challenges and opportunities in implementing TinyML models on IoT devices,

equipping them for real-world applications and innovations.

Looking ahead, there are several avenues for expanding and enhancing the course and its curriculum. One of the pivotal enhancements involves enlarging the number of categories in the emotion dataset to make it more comprehensive. Human emotions are diverse and extend beyond neutral and smiling states to include expressions like anger, sadness, fear, and more. By incorporating a richer and more varied dataset, students will be able to develop more sophisticated and accurate emotion recognition models. Furthermore, integrating newer IoT devices and technologies, as well as updating the course with state-of-the-art TinyML models and techniques, would ensure that the curriculum remains abreast with the latest advancements in the field.

Another essential aspect for future work is the establishment of collaborations with industry and academia. This would not only provide students with real-world datasets and challenges but also enable them to contribute to ongoing research and development in emotion recognition and IoT. In addition, fostering partnerships could facilitate guest lectures and workshops by experts in the field, thereby enriching the educational experience. As technology continues to evolve rapidly, creating a dynamic, adaptive, and collaborative learning environment that integrates diverse emotion datasets and stays in sync with technological advancements and industry trends is fundamental in cultivating well-rounded professionals in the domains of TinyML and IoT.

9. REFERENCES

Ankan Bhattacharyya, et al. (2021). A deep learning model for classifying human facial expressions from infrared thermal images. Sci Rep 11, 20696 (2021). https://doi.org/10.1038/s41598-021-99998-z

Arduino, I. (2019). Arduino IDE. https://www.arduino.cc/en/software

Alexander S. Gillis. (2022). Artificial intelligence of things (AIoT), URL: https://www.techtarget.com/iotagenda/definition/Artificial-Intelligence-of-Things-AIoT

Brian Plancher, et al. (2022) TinyMLedu: The Tiny Machine Learning Open Education Initiative. SIGCSE 2022: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education.

Egham UK. (2017). Gartner says 8.4 billion connected "things" will be in use in 2017, Up 31 percent from 2016. Gartner Inc.

ESP-FACE. (2019). Esp-face. https://github.com/Yuri-R-Studio/esp-face

FLICKR. (2023). Flickr. https://www.flickr.com/

Frahim J, Pignataro C, Apcar J, Morrow M. (2015) Securing the internet of things: A proposed framework. Cisco White Paper.

GOOGLE. (2023). Google images. https://www.google.com/imghp/

Harvard Tiny Machine Learning Open Education Initiative. (2022). Tiny Machine Learning Open Education Initiative (TinyMLedu), URL: https://tinyml.seas.harvard.edu/

Haoyu Ren, et al. (2021). TinyOL: TinyML with Online-Learning on Microcontrollers. International Joint Conference on Neural Network (IJCNN).

ITU-T Y. (2012). 2060: Overview of the Internet of things. ITU-T–International Telecommunication Union.

Ji Lin, et al. (2020) MCUNet: Tiny Deep Learning on IoT Devices. Computer Vision and Pattern Recognition, NeurIPS.

Ji Lin, et al. (2022). On-Device Training Under 256KB Memory, MIT, MIT-IBM Watson AI Lab.

Kaipeng Zhang, et al. (2016). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. Computing Research Repository (CoRR).

Kwon, et al. (2020). A Low-Power Hardware Platform for Implementing Tiny Machine Learning Systems for Real-Time Fall Detection. Sensors.

Mauro Conti, Ali Dehghantanha, Katrin Franke, Steve Watson. (2018) Internet of Things Security and Forensics: Challenges and Opportunities, (Elsevier) Future Generation Computer Systems Journal.

ISSN: 2473-4901
v9 n5973

Mark Sandler, et al. (2018). Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. Computing Research Repository (CoRR).

NVIDIA. (2023). High Performance Computing Products and Solutions, URL: https://www.nvidia.com/en-us/high-performance-computing/

Olga Russakovsky, et al. (2015) ImageNet Large Scale Visual Recognition Challenge. IJCV.

Partha Pratim Ray, et al. (2022) A review on TinyML: State-of-the-art and prospects, Journal of King Saud University - Computer and Information Sciences.

Peña-López I. (2005). ITU Internet report 2005: the internet of things.

Simone Disabato, et al. (2020) Incremental On-Device Tiny Machine Learning, AIChallengeIoT '20: Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things.

Servida F, Casey E. (2019) IoT forensic challenges and opportunities for digital traces. Digital Investigation.

TENSORFLOW. (2023). TensorFlow. https://www.tensorflow.org/

Xiaofeng Lu. (2022). Deep Learning Based Emotion Recognition and Visualization of Figural Representation. Front Psychol. 2022 Jan 6;12:818833. doi: 10.3389/fpsyg.2021.818833.

Yang, et al. (2021). Performance evaluation of tiny machine learning algorithms for human activity recognition on embedded devices. IEEE Internet of Things Journal.